



Hardware Realization of an FPGA Processor - Operating System Call Offload and Experiences

Hindborg, Andreas Erik; Karlsson, Sven

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Hindborg, A. E., & Karlsson, S. (2014). *Hardware Realization of an FPGA Processor - Operating System Call Offload and Experiences*. Poster session presented at 10th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, ACACES 2014, Fiuggi, Italy.

General rights

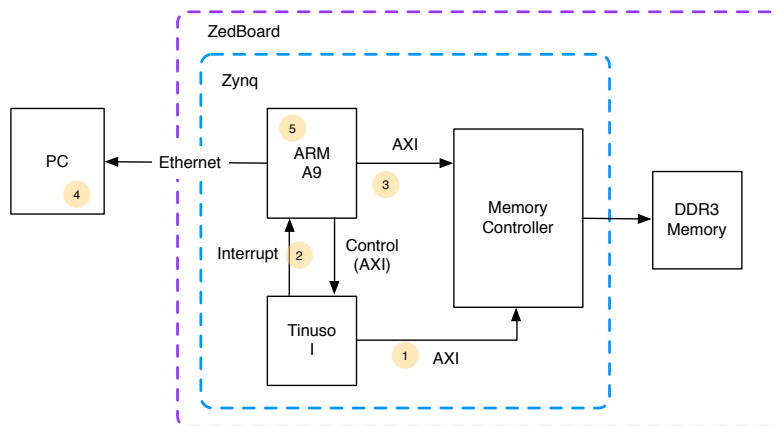
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

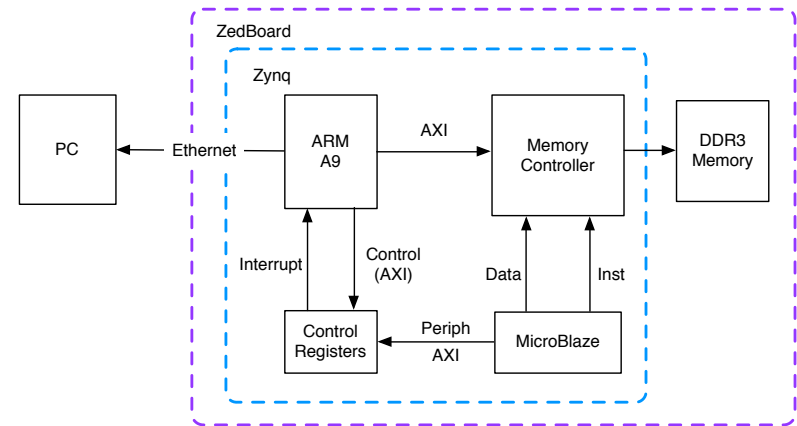
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Hardware Realization of an FPGA Processor - Operating System Call Offload and Experiences

Andreas Erik Hindborg and Sven Karlsson
Technical University of Denmark



Tinuso-I system



Baseline MicroBlaze system

Motivation

- Benchmark programs such as those found in SPEC and SPLASH require access to operating system services such as the file system in order to run.
- These services are often unavailable on embedded platforms, because it is unfeasible to run a full operating system on these platforms.
- It is desirable to provide access to these services in order to be able to benchmark the platform with standard benchmarks.
- Porting a full operating system is usually not feasible.

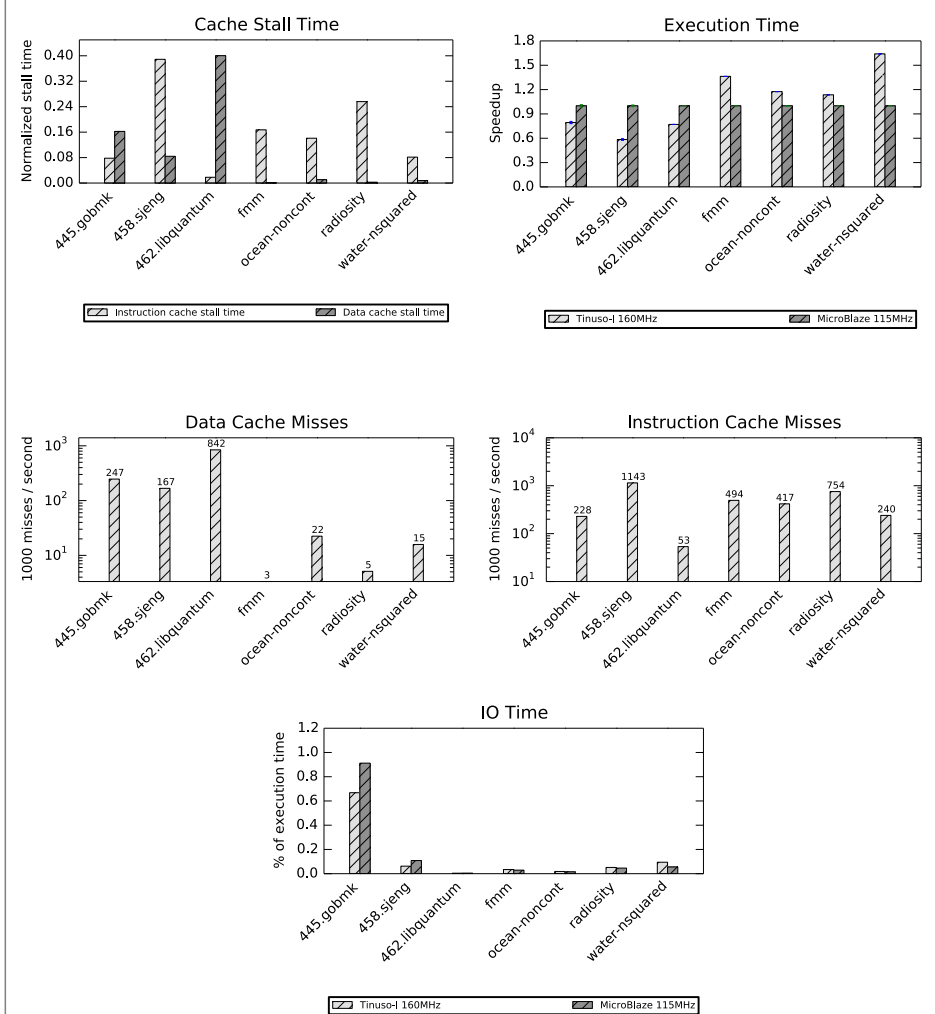
Contributions

- We propose and implement a method for offloading operating system services and demonstrate the method with a Tinuso-I based system and a MicroBlaze system.
- We demonstrate the proposed method by using it to evaluate the Tinuso-I soft core processor using industry standard benchmark applications.

Implementation

- Several software layers and components are needed to execute real applications. Applications are commonly developed assuming a POSIX compliant operating system. However, running a full operating system on small embedded systems is often unfeasible.
- We intercept file system service requests by linking the benchmark applications with a custom run-time library. When an application requests a file system service, such as invoking the open system call, the run-time library sends the request to a PC via a communication link. The response is received by the run-time and relayed back to the requesting program.
- With this approach it is possible to provide file system access to a processor core by only implementing the following: a) A minimal run-time library that intercepts file system service requests from programs running on the Tinuso-I core in FPGA fabric. b) Communication link support on the PC that services the file system requests and on the soft core. c) A service on the PC that responds to the file system service requests.
- We implement the proposed offloading method for a Tinuso-I core synthesised to the FPGA fabric of a Xilinx XC7Z020-CLG484-1 device. The silicon device is part of an AvNet ZedBoard development kit.

Results



Conclusion

- We use the Xilinx Zynq SoC to realise the Tinuso-I core and perform a hardware bring-up.
- We propose a method for offloading operating system service using the ARM core in the Xilinx Zynq SoC.
- We demonstrate our method by using it to evaluate both Tinuso-I and Xilinx MicroBlaze.
- We evaluate the system by executing a set of SPEC 2006 and SPLASH-2 benchmarks.
- We demonstrate a speedup of up to 64% over a similar Xilinx MicroBlaze baseline system.
- On average Tinuso-I performs 6% better than MicroBlaze while consuming 27% fewer LUTs.